



TITLE:

Serializable Classesの構造について (計算機構に関する数学的基礎理論 とその応用)

AUTHOR(S):

茨木, 俊秀; 亀田, 恒彦; 箕浦, 敏美

CITATION:

茨木, 俊秀 ...[et al]. Serializable Classesの構造について(計算機構に関する数学的基礎理論とその応用). 数理解析研究所講究録 1983, 494: 102-113

ISSUE DATE:

1983-06

URL:

<http://hdl.handle.net/2433/103573>

RIGHT:

Serializable Classes の構造について

京都大学工学部 茨木 俊秀

Simon Fraser 大 亀田 恒彦

Oregon State 大 箕浦 敏美

1. まえがき

データベース管理システム (DBMS) における同時処理制御 (concurrency control) は, 基本的な問題の一つとして, さまざまな立場から研究されている. 同時処理の妥当性の判定には serializability の概念が主として用いられるが, その一般的判定問題は NP 完全であって [4], 効率良いスケジューラが存在するとは考えられない. そこで, serializability の定義にいくつかの付加条件を加えることで, 判定が多項式時間で可能であるようなクラスの構成が試みられている. そのようなクラスの例に Papadimitriou による DSR [4] (Bernstein ら [1] の CPSR と同一) がある.

DSR は, 与えられたスケジュール内の operations の順序のある部分を保存するという付加条件によって定義される

が、この考え方を延長すると、保存すべき順序部分をどう定義するかで、種々のクラスが定義できる。ここでは、同一の data item を「読む」および「書く」という operations に関して定義されるいくつかの順序制約を導入し、得られたクラス間の包含関係を明らかにした。また、各クラスの所属問題が多項式時間のアルゴリズムをもつか、あるいは NP 完全になるかどうかを示した。

以上のクラスの多くは新しく定義されたものであるが、とくに、DSR を真に含み、しかも多項式時間アルゴリズムをもつクラス WR+RW の存在は注目に値しよう。また、これらの議論は、transaction input/output graph (TIO graph) 上の span nonoverlapping topological sort (SNOTS) という概念を用いて統一的に行われる。従来、serializability の議論が、クラス毎に異なる道具を用いてなされていたため、全体の見通しが悪いという欠点を有していたことに対し、同一の道具を一貫して用いることで、構造上の相違を際立たせることに成功したと思われる。

2. 諸定義

DBMS に transactions T_1, T_2, \dots, T_n が入力される。各 transaction はいくつかの read operations と write

operations から成る。ただし, data item X への read operation を $R_j[X]$, write operation を $W_j[X]$ と記す。 j は transaction T_j を示す添字である。 transactions に対しては, 一般モデルと 2ステップモデルがよく用いられる。 一般モデルでは, T_j を半順序の定義されたいくつかの $R_j[X]$ と $W_j[X]$ の集合と捉え (items X は種々存在する), 2ステップモデルでは, $R_j[X, Y, \dots]$ に $W_j[Z, \dots]$ が後続すると考へる。 $R_j[X, Y, \dots]$ は items X, Y, \dots への read operation であり, $W_j[Z, \dots]$ も同様に解釈する。 本論文の結果は, どちらのモデルに対しても成立する。

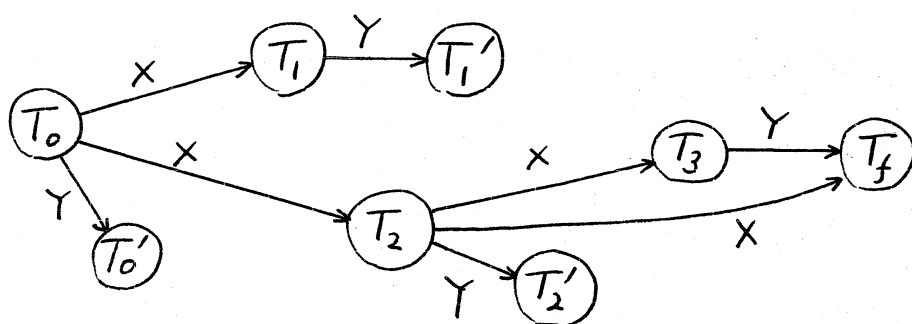
T_1, T_2, \dots, T_n に含まれる operations は, あらかじめ各 T_j 内で指定された順序を除けば, 勝手な順序で DBMS に到着する。 到着順に operations を並べたものを schedule という。 たとえば,

$$S = W_0[X, Y] R_1[X] R_2[X] W_2[X, Y] R_3[X] W_1[Y] \\ W_3[Y] R_f[X, Y] \quad (1)$$

である。 ここに, 最初の W_0 と最後の R_f は形式的に付されたもので, それぞれ最初にすべての items を書き, 最後にはすべての items を読む operations である。

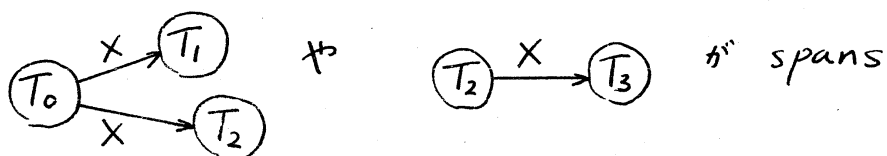
通常の DBMS では, 各 item のコピーを常に 1 個だけ保持する (backup 用は別にして) ので, $R_j[X]$ という

operation は, s においてその operation 以前に書かれた最新の item X を読むこととなる. このようなモデルを single version モデルという. スケジュール s に対し, グラフ $TIO(s)$ をつぎのように定義する. $TIO(s)$ は各 transaction T_j を節点とし, $W_k[X]$ で書かれた X を $R_i[X]$ が読むとき, ラベル X をもつ枝 (T_k, T_i) を作る. $W_k[X]$ で書かれた X が, どの read operation によっても読まれないとき, $W_k[X]$ を useless であるといい, T_k にダミー-節点 T'_k を付し, ラベル X をもつ枝 (T_k, T'_k) を書く. 式 (1) の s に対する $TIO(s)$ は

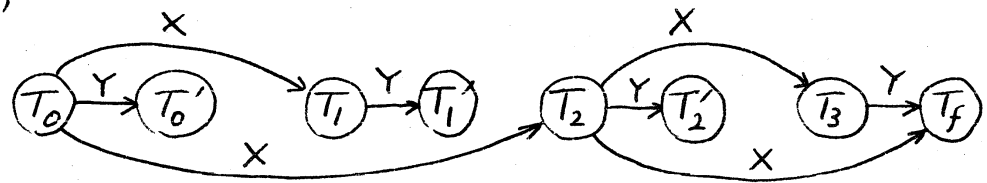


である. ある節点から発し, 同じラベルをもつ枝の集合を span という. $TIO(s)$ の全節点を左から右に, 枝方向に矛盾しないように一列に並べるとき, 同じラベルをもつ 2 個以上の spans が重ならないならば, SNOTS であるという.

上の例では



であり,



は SNOTS である.

スケジュール S と S' において, $TIO(S)$ と $TIO(S')$ が一致するならば, S と S' は互いに等価であるという. スケジュール S' において, operations が transaction 毎に整列しているとき, つまり (添字は $1, 2, \dots, n$ の順でなくてもいいが)

$$S' = W_0 R_1 W_1 R_2 W_2 \cdots R_n W_n R_f$$

という形をとるとき, serial スケジュールと呼ぶ. スケジュール S が, ある serial スケジュール S' に等価ならば, S は serializable であるという. serializable なスケジュールはデータ内の首尾一貫性を保持することが知られており, concurrent に処理することの妥当性を示すものである.

以下, serializable なスケジュールのクラスを SR と記す.

つぎの定理は, 以下の議論において基本的である.

定理 1. $S \in SR$ の必要十分条件は, $TIO(S)$ が acyclic かつ SNOTS をもつことである. \square

式 (1) の S は, 上の SNOTS から導びかれるように, serial スケジュール

$$S' = W_0 [X, Y] R_1 [X] W_1 [Y] R_2 [X] W_2 [X, Y] R_3 [X] W_3 [Y] R_f [X, Y]$$

に等価であるから, $s \in SR$ を得る.

3. 付加条件

スケジュール s において W_i が R_j に先行するとき

$$W_i \ll_s R_j$$

と書く. 他の組合せについても同様である. serializability の定義において, 等価となる serial スケジュール s' に関しつぎの付加条件を考える.

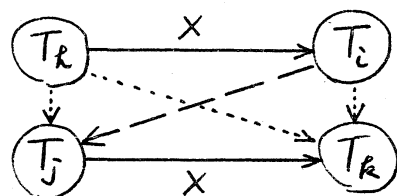
WW-条件: ある item X に対し, $W_i[X] \ll_s W_j[X]$

ならば, s' において T_i は T_j に先行する.

同様に, WR-条件, RW-条件, RR-条件も定義しよう. WW-条件の下で serializable なスケジュールのクラスを WW と記す. WR, RW, RR 等のクラスも同様である. つぎに, クラス $WR+RW$ は, WR-条件と RW-条件の両方をみたす s' に対し serializable であるようなスケジュールのクラスを記す. 他の組合せについても同様に定義する.

付加条件に対応して, グラフ $TIO(s)$ に条件枝を追加する. たとえば, WW-条件の場合, $W_i[X] \ll_s W_j[X]$ ならば, 枝 (T_i, T_j) を加えるわけである. これらの枝と SNOTS 条件から, さらに枝が追加される. 次図の状況において (T_i と T_k はグミ-節点であ, てもよい), 点線の枝の一つでも加われ

ば、破線の枝が追加され、 T_i を T_j に先行させるのである（そうでなければ SNOTS が存在しない）。

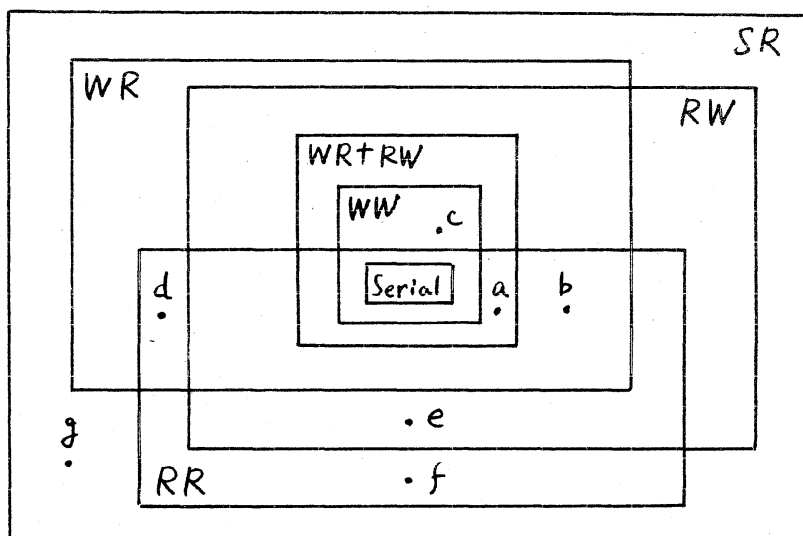


これは Sethi [5] のいう exclusion rule に本質的に同一である。得られたグラフを $TIO_{WW}(S)$ と呼ぶ。他のクラスについても同様に定める。

定理 2. 各クラス $* = WW, WR, RW, \dots$ などに対し, $S \in *$ の必要十分条件は, $TIO_*(S)$ が acyclic かつ SNOTS をもつことである。□

5. クラス間の包含関係

上記の $WW, WR+RW$ などのクラスの間の包含関係は下図



の如くである。とくに,

$$WW = WW + WR + RW = WW \cap WR \cap RW$$

$$WW \subseteq WR + RW \subseteq WR \cap RW$$

などが興味深い。前述の DSR は本来 $WW + WR + RW$ として定義されているが, この結果から WW に等しいことが判明した。図中のスケジュール a, b 等は以下の通りである。

$$a = W_0[x, Y] R_1[x] R_2[x] W_2[x, Y] R_3[x]$$

$$W_1[Y] W_3[Y] R_f[x, Y]$$

$$b = W_0[x] W_1[x] R_2[x] W_3[x] W_2[x] R_4[x] W_5[x] R_f[x]$$

$$c = W_0[x] R_2[x] R_1[x] W_2[x] R_f[x]$$

$$d = W_0[x] R_3[x] W_1[x] R_2[x] W_3[x] W_2[x] R_f[x]$$

$$e = W_0[x, Y] R_2[Y] R_1[x] W_2[x] W_1[x]$$

$$R_3[x] W_4[x] R_f[x, Y]$$

$$f = T_0 d^* T_1 e^* T_f$$

$$g = T_0 d^* T_1 e^* T_2 c^* T_f$$

ただし, T_1 と T_2 はすべての items を読んで書く transactions, d^* は d から W_0 と R_f を除いたもの。 e^*, c^* も同様である。

6. 特別な場合の包含関係

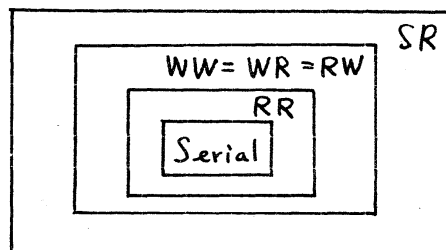
transactions の同時処理に際して, 以下の制約がしばし

ば付される。

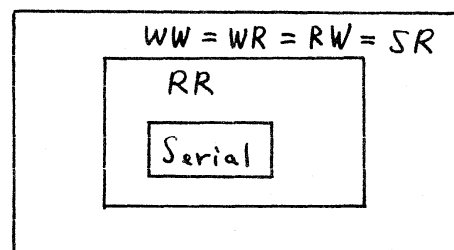
(1) useless operations W_j は存在しない。

(2) T_j が item X を書くならば, X を必ず読んでいなければならない。

これらの制約の下では, クラス間の包含関係は下図のように単純化される。



制約 (1)



制約 (2)

7. 多項式時間アルゴリズムをもつクラス

各クラスの所属問題に関連して, つぎの結果が得られる。

定理 3. $S \in WW$, $S \in WR + RW$ はいずれも多項式時間で判定可能である。□

$WW (=DSR)$ については [4] などに証明がある。 $WR + RW$ は WW を真に含み, しかも所属問題が多項式時間で判定可能であるのは興味深い。

$TIO(S)$ における $SNOTS$ に基づく定理 2 を用いると, これらの判定アルゴリズムを統一的に記述できる。すなわち, 4 節の $TIO_{WW}(S)$ および $TIO_{WR+RW}(S)$ を多項式時間で構成す

ると、これらが acyclic であるとき、かつその時に限り $S \in WW$ および $S \in WR+RW$ となる。acyclicity はよく知られているように、多項式時間で判定可能である。サイクルの有無に帰される理由は、 WW の場合、 WW -条件の枝と exclusion rule の枝を考慮すると、同一の item X を読み書きするすべての transactions 間に total order が定義されるので、 $SNOTS$ の判定が自明となるからである。 $WR+RW$ については、 T_i と T_j の両者とも useless X を書く場合を除いて、やはり順序が定まる。useless X を書く T_i と T_j はどちらを先にしても $SNOTS$ の存在に影響を与えないので、 $SNOTS$ の判定はやはり自明になされる。

8. NP 完全であるクラス

WW と $WR+RW$ 以外については、次の結果がある。

定理 4. クラス $SR, WR, RW, RR, WR+RR, RW+RR$ の所属問題はすべて NP 完全である。□

SR に関しては [4] に証明があるが、他は新しい結果である。いずれも充足可能性問題（詳しくは NOT-ALL-EQUAL SAT [2]）を、対応するクラスの $TIO_*(S)$ に帰着することで証明される。

9. Multi-Version モデルへの拡張

これまで、各 item のコピーは常に1個だけ存在するという前提の下に話を進めてきた。複数個（一般にはいくつでも）のコピーの存在を許せば、operation $R_j[X]$ に際して、それまでに書かれているコピーの中から適当なものを選んで読むことができる。この結果、これまでの定義では serializable ではないスケジュールも、multi-version の下では serializable となる可能性があり、同時処理の機能を高めることができる。本論文で導入したグラフ $TIO(S)$ と $SNOTS'$ という概念は、multi-version へも自然に拡張され、同種の議論を展開できる。得られた結果は別の機会に報告したい。

なお、本論文の定理の証明など、詳しい内容は、文献[3]を参照されたい。

文献

[1] P. A. Bernstein, D. W. Shipman and W. S. Wong, Formal aspects of serializability in database concurrency control, IEEE Trans. Software Eng. SE-5, 3, pp. 203-215, May 1979.

[2] M. R. Garey and D. S. Johnson, Computers and
—11—

Intractability: A Guide to the Theory of NP-completeness, Freeman, 1979.

[3] T. Ibaraki, T. Kameda and T. Minoura, *SNOTS* and its applications: Serializability theory made simple, Simon Fraser University, Department of Computing Science, TR 82-12, December 1982.

[4] C. H. Papadimitriou, The serializability of concurrent database updates, *JACM*, 26, 4, pp. 631-653, Oct. 1979.

[5] R. Sethi, A model of concurrent database transactions, *Proc. 22nd IEEE FOCS*, pp. 175-184, Oct. 1981.